arXiv:1211.3567v1 [math.NA] 15 Nov 2012

A Bernstein Polynomial Collocation Method for the Solution of Elliptic Boundary Value Problems

N. Mirkov*[1] and B. Rasuo[2]

[1]University of Belgrade, Institute of Nuclear Sciences 'Vinca', Serbia
[2]University of Belgrade, Faculty of Mechanical Engineering, Serbia

Abstract

In this article, a formulation of a point-collocation method in which the unknown function is approximated using global expansion in tensor product Bernstein polynomial basis is presented. Bernstein polynomials used in this study are defined over general interval $[a, b]$. Method incorporates several ideas that enable higher numerical efficiency compared to Bernstein polynomial methods that have been previously presented. The approach is illustrated by a solution of Poisson, Helmholtz and Biharmonic equations with Dirichlet and Neumann type boundary conditions. Comparisons with analytical solutions are given to demonstrate the accuracy and convergence properties of the current procedure. The method is implemented in an open-source code, and a library for manipulation of Bernstein polynomials bernstein-poly, developed by the authors.

1 Introduction

Bernstein polynomials are known because of their many useful properties [1]. When restricted to the unit interval they are used in the definition of Bézier curves, which are very important tools in computer graphics. On their own, they have enjoyed increased attention in recent years, specifically as means to represent solutions to differential equations. Bhatti and Bracken [2] present a Galerkin method which uses Bernstein polynomials as trial functions for solution of two-point boundary value problem, Yousefi et. al [3] use Bernstein polynomials in Ritz-Galerkin method to approximate solution of hyperbolic PDE with an integral condition. Doha et. al. [4] present the solution method for high-even-order ordinary differential equations.
Unlike all listed examples, which use Bernstein polynomials in some sort of

*Email: nmirkov@vinca.rs

Galerkin method, in this article we develope a collocation method. The motivation to do so comes from the nature of Bernstein polynomials themselves. When function is expanded in Bernstein polynomial basis, the expansion coefficients also respresent the nodal values of the expanded function. For two-point boundary value problems this leads to very simple implementation, allowing direct imposition of the boundary conditions at the end-points of the interval. When the unknown function is approximated as a global expansion in Bernstein polynomials basis, and used in a point-collocation method, the approach becomes very similar to pseudospectral methods [5], [6] [7]. Unlike their polynomial counterparts used in pseudospectral methods (e.g. Chebyshev and Legandre), Bernstein polynomials are not orthogonal, and many examples of basis transformation exist when this is necessary (eg. [8]).
For the present article we assume that L, the elliptic differential operator in interior, and B, the boundary operator are linear, and that they define well posed boundary value problem

$$L[u](\mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega. \tag{1}$$

$$B[u](\mathbf{x}) = g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega. \tag{2}$$

Where $\Omega \in \mathbb{R}^2$ is a given rectangular domain.
In the remainder of the paper we give a brief review of Bernstein polynomial properties, relevant to the proposed method. The solutions to PDEs defined over two-dimensional domains can be represented by surfaces embedded in three-dimensional Eucledian space. This serves as rationale of Section 3, in which we describe surfaces defined by expansions in tensor product Bernstein polynomial basis, also giving expressions for elliptic operators that we will use subsequently. Then, in Section 4, we describe the fomulation of the proposed collocation method. Finally we give couple of example solutions, as well as their error, and convergence analysis in Section 5.

2 Properties of Bernstein Polynomials

In further discussion we consider only generalized Bernstein polynomials, those defined over arbitrary interval $[a, b]$, and simply call them Bernstein polynomials. The Bernstein polynomials of nth degree form a complete basis over $[a, b]$, and are defined by

$$B_{i,n}(x) = \binom{n}{i} \frac{(x-a)^i (b-x)^n - i}{(b-a)^n}, \quad i = 0, 1, ..., n, \tag{3}$$

where binomial coefficients are given by

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}. \tag{4}$$

They satisfy symmetry $B_{i,n}(x) = B_{n-i,n}(1-x)$, positivity $B_{i,n}(x) \geq 0$, and they form partition of unity $\sum_{i=0}^{n} B_{i,n}(x) = 1$ on defining interval $x \in [a, b]$. For

$i = 0$, and $i = n$, they have value equalt to one at $x = a$, and $x = b$, respectively. Otherwise, they have a unique local maximum occuring at $x = i/n$, having the value $B_{i,n}(i/n) = i^i n^{-n}(n-i)^{n-i}\binom{n}{i}$.
In a recent article [4], authors present explicit formula for calculation of arbitrary order derivatives of Bernstein basis functions of any order defined over standard interval $[0, 1]$. In an atempt to formulate collocation method for solution of BVP's over arbitary intervals, first step would be to write down this expression for Bernstein polynomials defined for arbitrary interval.

Let $B_{i,n}(x)$ define $i-th$ basis function of $n-th$ order, at point x, where $x \in [a, b]$, and $i \in [0, n]$. Then, the derivative of order p of such basis function can be expressed as:

$$D^p B_{i,n}(x) = \frac{n!}{(n-p!)(b-a)^p} \sum_{k=max(0,i+p-n)}^{min(i,p)} \binom{p}{k} B_{i-k,n-p}(x)$$

Useful proprerty of Bernstein basis functions is that they all vanish at endpoints of the interval, except the first and the last one, which are equal to one at $x = a$ and $x = b$ respectively.

The factorial formula Eq. 4 for binomial coefficients is known not to be most numerically efficient for large numbers, and the alternative representations exist, most numerically efficient being the multiplicative formula, which we will use here

$$\binom{n}{i} = \prod_{i=1}^{k} \frac{n-(k-i)}{i}. \tag{5}$$

3 Surfaces defined by tensor product of Bernstein polynomials

Following equations will help us in developing a collocation method using Bernstein polynomials for boundary value problems defined over two-dimensional domains.

Let the following equation represent a surface in $\mathbb{R}^3$ defined by a tensor product of Bernstein basis functions

$$f(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{m} \beta_{i,j} B_{i,n}(x) B_{j,m}(y). \tag{6}$$

We can define p-th order partial derivative in x -axis direction in a following way

$$\frac{\partial^p f(x, y)}{\partial x^p} = \sum_{i=0}^{n} \sum_{j=0}^{m} \beta_{i,j} D^p B_{i,n}(x) B_{j,m}(y), \tag{7}$$

and an q-th order derivative in y -axis direction

$$\frac{\partial^q f(x,y)}{\partial y^q} = \sum_{i=0}^{n}\sum_{j=0}^{m} \beta_{i,j} B_{i,n}(x) D^q B_{j,m}(y). \tag{8}$$

Finally mixed derivative of order $p+q$ is defined as follows:

$$\frac{\partial^{p+q} f(x,y)}{\partial x^p y^q} = \sum_{i=0}^{n}\sum_{j=0}^{m} \beta_{i,j} D^p B_{i,n}(x) D^q B_{j,m}(y). \tag{9}$$

This paper deals with elliptic equations, therefore we will define two differential operators, most common in second and forth order boundary value problems. Laplacian operator can be written as

$$\Delta f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} =$$

$$= \sum_{i=0}^{n}\sum_{j=0}^{m} \beta_{i,j} B_{j,m}(y) D^2 B_{i,n}(x) + \sum_{i=0}^{n}\sum_{j=0}^{m} \beta_{i,j} B_{i,n}(x) D^2 B_{j,m}, \tag{10}$$

which can further be simplified to

$$\Delta f(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} \beta_{i,j} \left[B_{j,m}(y) D^2 B_{i,n}(x) + B_{i,n}(x) D^2 B_{j,m}(y) \right]. \tag{11}$$

Bicharmonic operator or bilaplacian can be written as

$$\Delta^2 f(x,y) = \frac{\partial^4 f(x,y)}{\partial x^4} + 2\frac{\partial^4 f(x,y)}{\partial x^2 \partial y^2} + \frac{\partial^4 f(x,y)}{\partial y^4} =$$

$$= \sum_{i=0}^{n}\sum_{j=0}^{m} \beta_{i,j} \left[B_{j,m}(y) D^4 B_{i,n}(x) + 2 D^2 B_{i,n}(x) D^2 B_{j,m}(y) + B_{i,n}(x) D^4 B_{j,m}(y) \right]. \tag{12}$$

4 Collocation method formulation

In present method, it is assumed that a variable can be expressed as an approximation in the form of global expansion in tensor product Bernstein polynomial basis.

We look for the approximate solution in the form (6) for $(x,y) \in \bar{\Omega} = \Omega \cup \partial\Omega$. The expansion coefficients $\beta_{i,j}$ are unknown, and need to be determined.

The collocation method, that we use to find the unknown coefficients, is a numerical procedure in which we require that solution satisfies differential equation exactly in a discrete set of points, known as collocation points. Number of collocation points has to match the number of unknowns.

If the order of Bernstein polynomial is n and m in x and y-axis directions respectively, then there is $n_c = (n+1) \times (m+1)$ unknown coefficients in global polynomial expansion. If we are solving homogenous Dirichlet problem the number of unknown is $n_{cd} = (n-1) \times (m-1)$. Reason for this is a property of Bernstein polynomials that we take the advantage of, namely the property to vanish at end points, except for the first and the last one, that we mentioned in Section 2. If the function value at the boundary is equal to zero, we eliminate the non-vanishing basis functions by setting their associated expansion coefficients to zero, and continue by solving the problem for the smaller number of unknowns n_{cd}.
Discretized equation set is obtained by substituting the unknown function and it's derivatives with the appropriate representation in Bernstein polyniomial basis, as described in Section . Substituting these Bernstein polynomial interpolants into the original equation set, and applying it at the set of collocation points, one arrives at a linear system of equations

$$Ab = c. \tag{13}$$

An element of the system matrix $a_{i,j}$ is obtained by evaluating the differential operator on the tensor product of ith and jth Bernstein polynomial basis function, at the collocation point defined by Cartesian coordinate pair (x_i, y_i).

$$a_{i,j} = L\left[B_{i,n}(x_i)B_{j,m}(y_i)\right], \quad i = 1,..,n-1, j = 1,..,m-1. \tag{14}$$

An element of the right-hand side vector c_i is an evaluation of the right-hand side function at the ith collocation point defined by Cartesian coordinates (x_i, y_i).

$$c_i = f(x_i, y_i), \quad i = 1, ..., n_{cd}. \tag{15}$$

For the cases other than those with homogenous Dirichlet boundary conditions, the number of unknowns, as being said, is larger and linear system has the additional rows defined by

$$a_{i,j} = B\left[B_{i,n}(x_i)B_{j,m}(y_i)\right], \tag{16}$$

$$c_i = g(x_i, y_i), \tag{17}$$

Where coordinates (x_i, y_i) define colocation points that belong to the domain boundary. Index pairs (i, j) belong to the set $(0, 1...m-1) \cup (n, 1...m-1) \cup (1...n-1, 0) \cup (1...n-1, m) \cup (0, 0) \cup (n, 0) \cup (0, m) \cup (n, m)$. There is in total $2(n+m)$ additional rows required to specify boundary conditions.
The change in boundary conditions (2), amounts to changing a few rows in the matrix A, as well as in the right-hand side vector c. Using expressions defined in Section 3. we may define any type of boundary conditions. In particular, the operator B takes the form of Eq.(6) when we need to impose Dirichlet boundary conditions, or (7),(8), with $p = q = 1$ for Neumann boundary conditions.
Solution vector b is as an auxilliary one-parameter array, it's values are moved to

two-parameter array of Bernstein polynomial expansion coefficient β. Mapping has the following form

$$\beta_{i,j} = b_{inode}, \tag{18}$$

Where $inode = (m+1) * (i-1) + j$, and $i = 0, ..., n, j = 0, ..., m$. If only inner expansion coefficients are sought for, in the case of homogenous Dirichlet problems, coefficient mapping takes the same form (18), but the indexes are defined by $inode = (m-1) * (i-1) + j - 1$, and $i = 1, ..., n-1, j = 1, ..., m-1$.

Finally we note that the distribution of points on a tensor product grid may be uniform, or non-uniform depending on situation. The full linear system 13 is solved using LU decomposition with partial pivoting.

5 Example Application

Collocation method formulated in previous section is implemented in bernstein-poly, a library for manipulation of Bernstein polynomials. It is an open-source code developed by the authors, written in Python [9]. In what follows, we will present couple of examples, with the purpose to illustrate validity and accuracy of the present collocation method. In all examples we use the same order of Bernstein polynomials in both x and y axis direction.

Example 1.

Consider Poisson equation

$$\Delta u(x, y) = f(x, y), \quad x, y \in [-1, 1], \tag{19}$$

with homogenous Dirichlet boundary conditions. Source term is defined as $f(x, y) = -2\pi^2 sin(\pi x)\, sin(\pi y)$. Exact solution for the given problem is

$$u_{exact} = sin(\pi x)\, sin(\pi y) \tag{20}$$

Fig. 1 shows numerical solution for approximation using Bernstein polynomials of order n=21. The absolute numerical solution error $abserr(i, j) = \|u(i, j) - u^e_{i,j}\|$, where $u^e_{i,j}$ represents the exact solution, is estimated at each (i, j)th collocation node, and the result is presented in Fig. 2. To study how accuracy changes with incresing the order of polynomial approximation we use L^2 relative error norm, defined in the following way

$$L^2 Error = \frac{\sum_{i,j=0}^{n,m} \left(u_{i,j} - u^e_{i,j}\right)^2}{\sum_{i,j=0}^{n,m} (u^e_{i,j})^2}. \tag{21}$$

Results of the analysis of L^2 relative error norm are sumarized in Table 1. To study order of accuracy it is useful to plot L^2 relative error norm as a function of polynomial order, n, which is done in *log-log* plot in Fig. 3. We observe exponential decay in the error for polynomial orders up to $n = 17$, after which the error decrease slows down. Eventually for order of Bernstein basis polynomials higher than $n = 21$, after which the error continually grows.

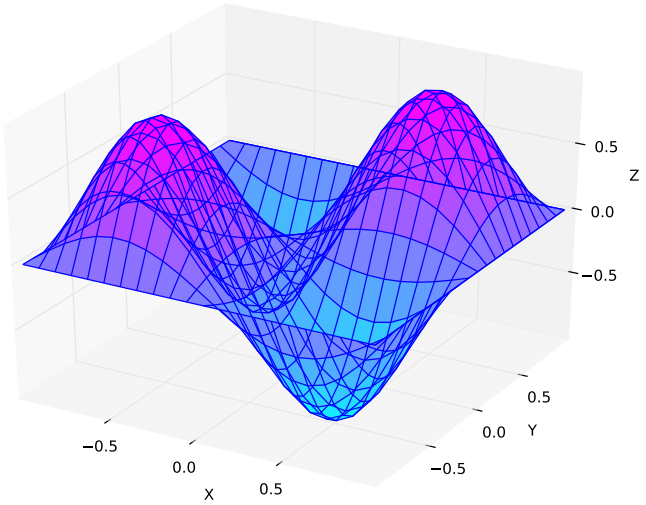

Figure 1: Bernstein polynomial collocation method solution of Poisson equation (Example 1), n=20.

Example 2.

In this example we consider the Poisson equation with non-homogenous Dirichlet boundary conditions. Treatment of this problem differs from the previous one algorithmically in a way we impose boundary conditions. For every collocation point at the boundary we write an additional linear equation and we solve linear system of size $(n_c) \times (n_c)$.
Source term in the Poisson equation (19) for this problem is defined by $f(x, y) = 6xy(1 - y) - 2x^3$, and solution domain by unit square $x, y \in [0, 1]$. The exact solution for this problem is

$$u_{exact} = y(1 - y)x^3 \tag{22}$$

The figures 4, and 5 show numerical solution and absolute error distribution for the case of $n = 20$. very high accuracy is achieved quite "early", for $n = 12$ it reaches order of 10^{-15}, as seen in Table 2.

Example 3.

Next example problem is defined by the Helmholtz equation

$$(\Delta + \lambda)u = f(x, y). \tag{23}$$

The problem is originally found in [10], and is defined by $\lambda = 1$, $f(x, y) = x$, and non-homogenous Dirichlet boundary conditions which are derived from the

n	L^2 rel. error
11	1.171×10^{-5}
13	3.170×10^{-7}
15	6.536×10^{-9}
17	1.049×10^{-10}
19	5.137×10^{-11}
21	3.111×10^{-11}
23	3.966×10^{-9}
30	3.146×10^{-08}
41	3.499×10^{-07}
51	1.091×10^{-05}
61	1.140×10^{-04}
71	1.346×10^{-04}

Table 1: L^2 relative error norm for the Example 1.

n	L^2 rel. error
12	5.841×10^{-15}
14	2.595×10^{-14}
16	1.754×10^{-13}
18	5.039×10^{-12}
20	1.544×10^{-10}
30	1.907×10^{-8}

Table 2: L^2 relative error norm for the Example 2.

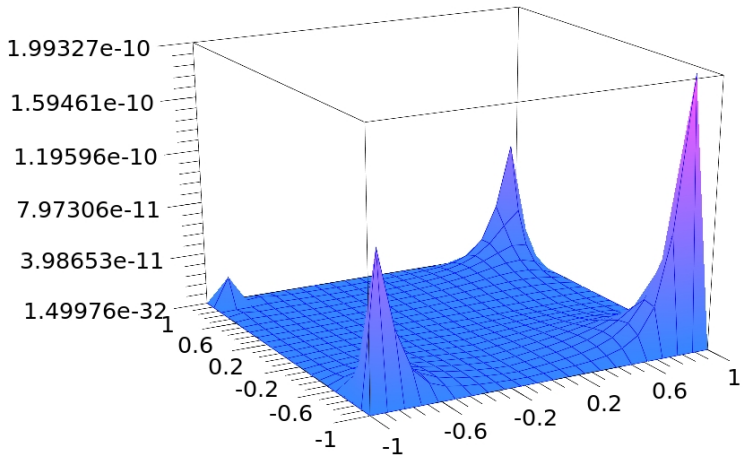

Figure 2: Absolute error (Example 1), n=20.

exact solution. Domain of solution is square $x, y \in [-\pi, \pi]$. Using Bernstein polynomials defined over general interval proves practical for this problem, as mapping of the domain to the unit square is not necessary.
This problem is exactly solvable and the solution is

$$u_{exact} = sin(x) + sin(y) + x. \tag{24}$$

As in previous examples we show numerical solution Fig. 6 and the distribution of the absolute error Fig. 7. Table 3 lists L^2 relative error norm variation with increasing order of polynomial basis.

Example 4.

In next example we give the solution to biharmonic equation, which is often encountered in the theory of ellasticity, as it describes deflections of loaded plates.

$$\Delta^2 u(x, y) = f(x, y). \tag{25}$$

Two types of boundary conditions can be defined for this problem (I) u and $\partial^2 u/\partial n^2$ or (II) u and $\partial u/\partial n$. Both cases are interesting on their own because

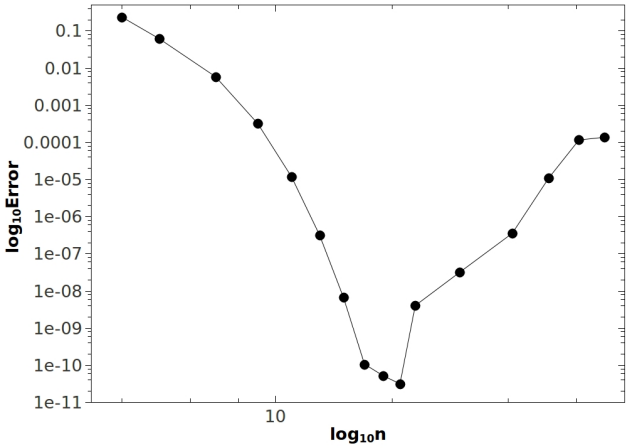

Figure 3: Example 1. L^2 relative error norm.

n	L^2 rel. error
12	9.035×10^{-6}
14	2.430×10^{-7}
16	4.992×10^{-9}
18	8.107×10^{-11}
20	4.057×10^{-11}
22	1.051×10^{-9}
30	1.533×10^{-7}

Table 3: L^2 relative error norm for the Example 3.

they require different solution approach. We will split discussion on biharmonic equation in two parts, present Example we will treat Type I and in the next one Type II boundary conditions.
In the case of Type I boundary conditions, biharmonic equation can be split into two coupled Poisson equations

$$\Delta v(x, y) = f(x, y), \tag{26}$$

$$\Delta u(x, y) = v(x, y). \tag{27}$$

The first example solution of biharmonic equation, taken from [11], will deal with the case of simply supported rectangular plate $0 \leq x \leq a, 0 \leq x \leq b$. Homogenous boundary conditions for both the function value and it's second derivatives in the direction normal to boundary are prescribed.
Source function, that describes load distribution over the surface in theory of

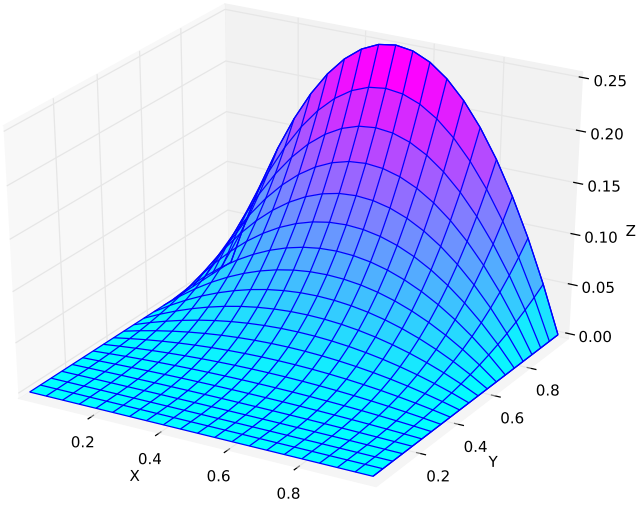

Figure 4: Solution of Poisson equation, non-homogenous Dirichlet BCs (Example 2), n=20.

plates is given by

$$f(x,y) = \pi^4 \left(\frac{m^2}{a^2} + \frac{n^2}{b^2} \right)^2 sin\frac{m\pi x}{a} sin\frac{n\pi y}{b}. \tag{28}$$

Biharmonic equation defined in such a way, allows the exact solution

$$u_{exact}(x,y) = sin\frac{m\pi x}{a} sin\frac{n\pi y}{b}. \tag{29}$$

For our purpose we set values $m = n = 1$, $a = b = 1$. Example solution for the order of polynomial, $n = 20$ is shown in Fig 8. The peaks in the absolute error near the corner nodes, as noticed in previous examples, is inherent to the present method, therefore we set an upper limit on the vertical axis here, to be able to get better picture of the distribution of absolute error in the rest of domain, Fig. 9. Table 4 shows variation in the error norm, in which the trend conforms to the one in previous examples.

Example 5

Biharmonic equation that is solved in this example has Type II boundary conditions and cannot be reformulated as a system of two coupled Poisson equations. This example is also taken from [11]. We will consider two cases, one having the exact solution

Case 5a:

$$f(x,y) = 56400(a^2 - 10ax + 15x^2)(b-y)^2y^4$$

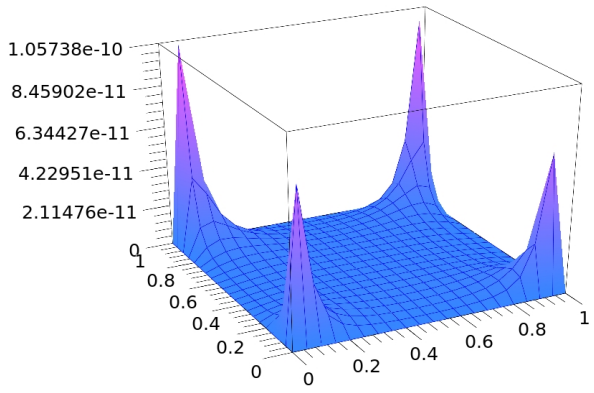

Figure 5: Absolute error (Example 2), n=20.

n	L^2 rel. error
10	6.538×10^{-8}
12	4.854×10^{-10}
14	2.832×10^{-12}
16	1.284×10^{-12}
20	9.467×10^{-11}
30	5.939×10^{-8}

Table 4: L^2 relative error norm for the Example 4.

$$+18800x^2(6a^2 - 20ax + 15x^2)y^2(6b^2 - 20by + 15y^2)$$
$$+ 56400(a - x)^2x^4(b^2 - 10by + 15y^2), \tag{30}$$

And one where the exact solution is unknown *Case 5b:*

$$f(x, y) = p_0 = const, \tag{31}$$

where p_0 takes value of 1000. The notation has physical significance, Example 5b represents the case of a plate clamped at all four sides, and exposed to uniform load of fluid pressure. Example 5a admits the exact solution

$$u_{exact}(x, y) = 2350x^4(x - a)^2y^4(y - b)^2. \tag{32}$$

In the assembling procedure, instead of writing the Eq. 14 for the nodes laying right next to the egde nodes, we use Eq. 16, which to remind once again, uses (x, y) values of the points on the boundary edge. These edge points are all

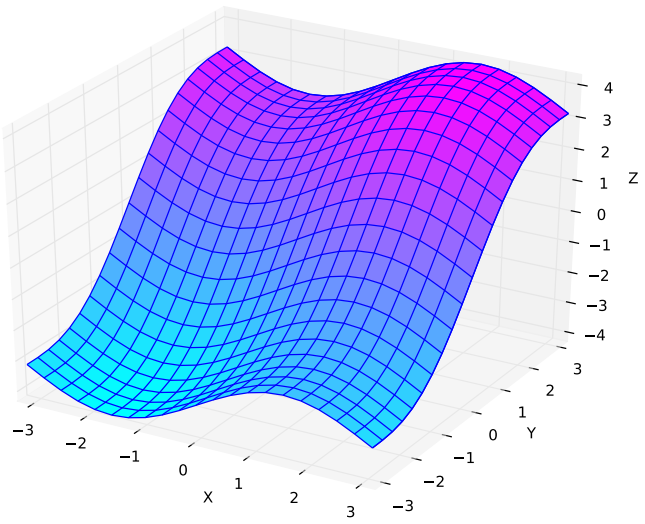

Figure 6: Solution of Helmholtz equation, non-homogenous Dirichlet BCs (Example 3), n=20.

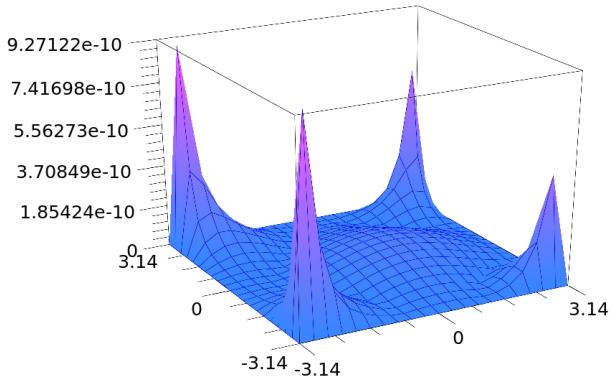

Figure 7: Absolute error (Example 3), n=20.

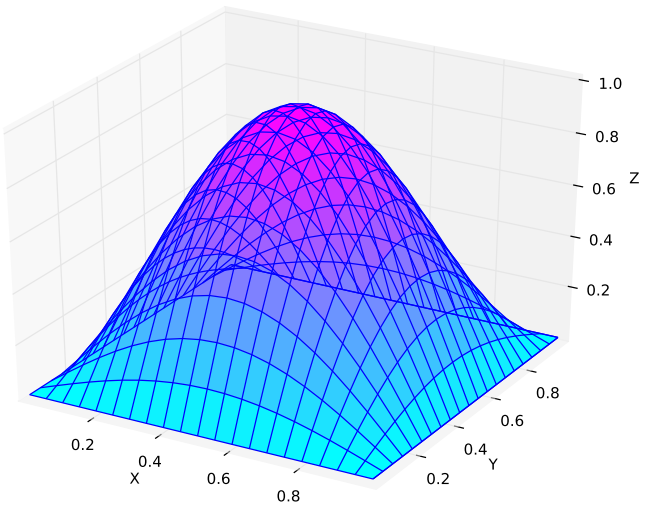

Figure 8: Solution of Biharmonic equation (Example 4), Type I BCs, n=20.

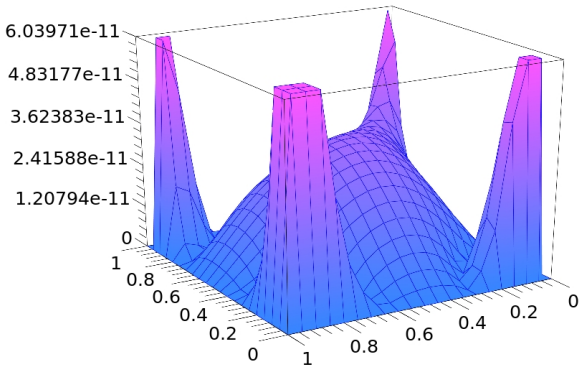

Figure 9: Absolute error (Example 4), n=20.

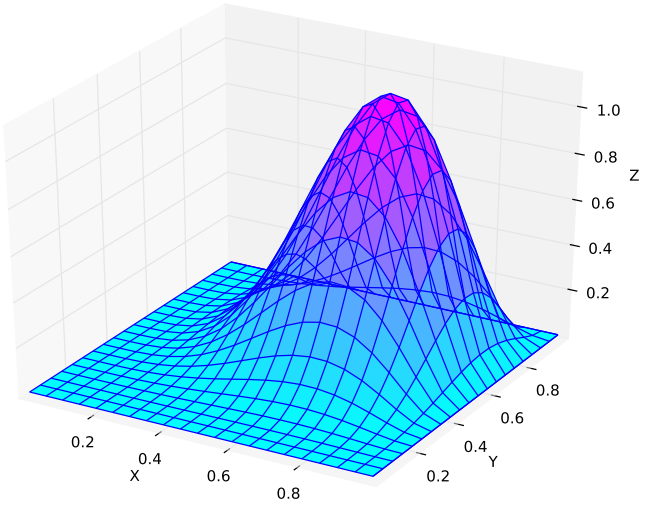

Figure 10: Solution of Biharmonic equation (Example 5a), Type II BCs, n=20.

n	L^2 rel. error
8	2.506×10^{-14}
10	1.064×10^{-14}
12	3.773×10^{-13}
14	1.174×10^{-11}
20	3.853×10^{-10}

Table 5: L^2 relative error norm for the Example 5a.

immediate neighbours of those collocations points, we would normally write the equation for. We note that the Neumann boundary conditions are not written for the corner nodes. Numerical results, consistent with previous examples, is shown in Fig. 10, Fig. 11 and Table 5. When there is no exact solution, we need to set up a criteria what solution to except as converged one. In all previous examples, which allowed exact solution, we see that the high-order accuracy is achieved by a comparatively small number of nodal points. For 21 nodal point in each direction (order of polynomial n=20), the L^2 relative error norm is of the order 10^{10} to 10^{11}. We should have the additional confidence in the result, if the spatial variation of solution expressed in terms of local maxima and minima, is not significant within the domain. Fig. 12 shows numerical solution with Bernstein polynomials or order n=20 in each direction.

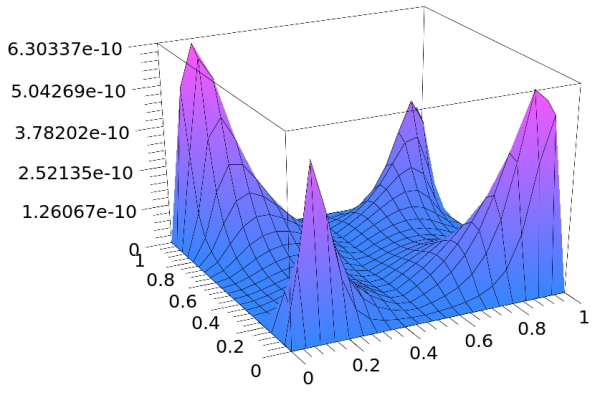

Figure 11: Absolute error (Example 5a), n=20.

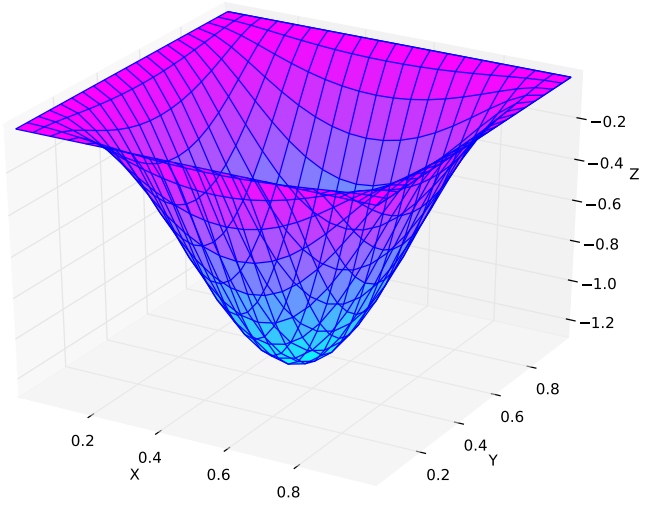

Figure 12: Solution of Biharmonic equation (Example 5b), Type II BCs, n=20.

6 Conclusions

In this paper a novel formulation of the collocation method using Bernstein polynomials is proposed. The main reason why the collocation method is chosen, are it's flexibility and simple implementation. The methodology presented in this paper has been implemented in bernstein-poly code [9], and several examples have been shown, where the elliptic boundary value problems in two dimensional domains were succesfully solved. Numerical results obtained by this method were compared with existing exact solutions. Excellent agreement and high accuracy is achievent even with small number of basis polynomials.
Trough extstensive testing we have concluded that the three components of the algorithm used here: defining polynomials over general interval, the non-recursive formulation for derivatives and the use of multiplicative formula for individual binomial coefficients significantly enhance capabilities of the present procedure related to the previous Bernstein polynomial methods in terms of the flexibility and speed.
As we have seen in log-log plots of L^2 relative error norm in all examples, the error decreases exponentially as the order of polynomial increases, until, around $n = 17 - 20$, when it changes the character and continually increases for higher values of n. We noticed the same character of error variation with n in all the examples above, which is related to the loss in accuracy in floating-point arithmetics with large numbers originating from factorials in the definition of Bernstein polynomials. This suggest further direction of investigation - developing a method based on a principle of domain decomposition. In that case in each "element" we would keep moderate degree in Bernstein polynomial, and the method would resemble spectral element method. Similar has been done e.q. [12], where Chebyshev multidomain method has been put forward. That study has served as a basis for development Spectral Difference Method. Another advantage of the domain decomosition approach is that it produces sparse matrices, with the sparsity pattern usual for tensor product grid discretizations. Future plan is to develop bernstein-poly to the point where highly accurate solution of Navier-Stokes equations in complex three-dimensional domains is possible.

Appendix A. Implementation of a new BVP in bernstein-poly

Setting up a new elliptic boundary value problem in bernstein-poly code [9] is briefly described here on the case of Poisson equation presented in the Example 3. Main program is located in collocation_test_2D.py, where the user needs to define solution domain by specifying it's x and y axis extents, and the order of approximating polynomials - n.

Listing 1: Setting up domain borders and approximating polynomial order.

```
# Bernstein polynomial order − the same order in both directions.
n = 12
m = 12
# The number of unknowns − all nodes counted for the case with non−homogenous BCs.
nvar = (n+1)*(m+1)
...
# Solution interval [x1,x2] x [y1,y2].
x1 = 0.
x2 = 1.

y1 = 0.
y2 = 1.
...
# Uniform mesh.
nd = linspace(x1,x2,n+1)
```

Specifying the problem generaly described by Eq. (1) is straight-forward:

Listing 2: Defining RHS vector entry

```
def rhs(x,y):
    return −(6*x*y*(1−y)−2*x**3)
```

Listing 3: Defining system matrix entry

```
def lhs(i,j,n,m,x1,x2,y1,y2,x,y):
    return −laplacian(i,j,n,m,x1,x2,y1,y2,x,y)
```

Listing 4: Defining the exact solution.

```
def exact_solution(x,y):
        return y*(1−y)*x**3
```

Boundary conditions have to be specified next. First we'll have a look how does the LHS matrix and RHS vector assembly look like (Listings 5 and 6).

Listing 5: Forming RHS vector in the main function.

```
f = zeros(nvar) # RHS vector
for i in range(0,n+1):
```

```
    x = nd[i]
    for j in range(0,m+1):
        y = nd[j]
        node = (m+1)*(i-1)+j
# Non-homogenous BCs...
        if (i==0):
# Left side:: Run trough all betas
            f[node] = bc_left(x,y)
        elif (j==0):
# Bottom:: Run trough all betas
            f[node] = bc_bottom(x,y)
        elif (i==n):
# Right: Run trough all betas
            f[node] = bc_right(x,y)
        elif (j==n):
# Top: Run trough all betas
            f[node] = bc_top(x,y)
        else:
            f[node] = rhs(x,y)
```

Listing 6: Forming system matrix

```
K = zeros( (nvar,nvar) ) # LHS matrix
for i in range(0,n+1):
    x = nd[i]
    for j in range(0,m+1):
        y = nd[j]
        node = (m+1)*(i-1)+j # node defines specific location on a grid
# Non-homogenous BCs...
        if (i==0):
# Left side:: Run trough all betas
            for k in range(0,n+1):
                for l in range(0,m+1):
                    jfun = (m+1)*(k-1)+l
                    K[node,jfun] = Dirichlet(l,k,n,m,x1,x2,y1,y2,x,y)
        elif (j==0):
# Bottom:: Run trough all betas
            for k in range(0,n+1):
                for l in range(0,m+1):
                    jfun = (m+1)*(k-1)+l
                    K[node,jfun] = Dirichlet(l,k,n,m,x1,x2,y1,y2,x,y)
        elif (i==n):
# Right: Run trough all betas
            for k in range(0,n+1):
                for l in range(0,m+1):
                    jfun = (m+1)*(k-1)+l
                    K[node,jfun] = Dirichlet(l,k,n,m,x1,x2,y1,y2,x,y)
        elif (j==n):
# Top: Run trough all betas
            for k in range(0,n+1):
```

```
                    for l in range(0,m+1):
                        jfun = (m+1)*(k-1)+l
                        K[node,jfun] = Dirichlet(l,k,n,m,x1,x2,y1,y2,x,y)
        else:
# Interior: Run trough all betas
                for k in range(0,n+1):
                    for l in range(0,m+1):
                        jfun = (m+1)*(k-1)+l
                        K[node,jfun] = lhs(l,k,n,m,x1,x2,y1,y2,x,y)
```

Listing 7: System matrix entry originating from the boundary operator (2)

```
def Dirichlet(i,j,n,m,x1,x2,y1,y2,x,y):
# Matrix entry for Dirichlet BCs:
    return basis_fun_eval(i,n,x1,x2,x)*basis_fun_eval(j,m,y1,y2,y)

def Neumann_y_dir(i,j,n,m,x1,x2,y1,y2,x,y):
# Matrix entry for Neumann BCs:
    return basis_fun_eval(i,n,x1,x2,x)*basis_fun_der(1,j,m,y1,y2,y)
```

RHS vector entry for the collocation point located at domain boundary is evaluated according to function $g(x)$ Eq. (2).

Listing 8: Evaluating RHS vector entry for the points on domain boundary.

```
def bc_right(x,y):
# Boundary condition for right edge of the rectangle
    return y*(1-y)
```

The beauty of the proposed method and it's Python implementation, is that non-trivial problems descibed using PDEs are solved in a very small number of command lines. The guiding principle behind the development of the code is modularity and escalation towards solution of problems with greater complexity. Readers are encouraged to use and upgrade bernstein-poly in their own research.

References

[1] G. G. Lorentz. *Bernstein Polynomials*. University of Toronto Press, Toronto, Canada, 1953.

[2] M. I. Bhatti and P. Bracken. Solutions of differential equations in a Bernstein polynomial basis. *Journal of Computational and Applied mathematics*, 205(1):272–280, 2007.

[3] S.A. Yousefi, Z. Barikbin, M. Denhgan. Bernstein Ritz-Galerkin method for solving an initial-boundary value problem that combines neumann and integral condition for the wave equation. *Numerical Methods for Partial Differential Equations*, 26:1236–1246, 2009.

[4] E. H. Doha, A. H. Bhrawy, M. A. Saker. On the derivatives of Bernstein polynomials: An application for the solution of high-even-order deifferential equations. *Boundary Value Problems*, 2011.

[5] J. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover, 2nd edition, 2001.

[6] B. Fornberg. *A Practical Guide to Pseudospectral Methods*. Cambridge University Press, 1998.

[7] M. Y.Hussaini, C. L.Streett, T. A.Zang. Spectral Methods for Partial Differential Equations. NASA Contractor Report, NASA-CR-172248, 1983.

[8] R. T. Farouki. Legendre-Bernstein basis transformations. *J. Comput. Appl. Math.* 119: 145–160, 2000.

[9] http://code.google.com/p/bernstein-poly/

[10] Y. C.Hon, W. Chen. Boundary knot method for 2D and 3D Helmholtz and convection-diffusion problems under complicated geometry. *International Journal of Numerical Methods in Engineering*, 56:1931–1948, 2003.

[11] M. Arad, A. Yakhot, G. Ben-Dor. A Highly Accurate Numerical Solution of a Biharmonic Equation. *Numerical Methods for Partial Differential Equations*, 13:375–391, 1997.

[12] D. A.Kopriva, J. H.Kolias. A conservative staggered-grid Chebyshev multidomain method for compressible flows. *Journal of Computational Physics*, 125(1):244–261, 1996.